Ravi S Ganti. Usage of Large Scale Data Processing for analyzing social sciences data. A Master's Paper for the M.S. in I.S degree.April, 2014. 38 pages. Advisor: Arcot Rajasekar

Social sciences data typically consists of a lot of surveys, each with multiple questions. These surveys are mostly in XML format with the size of the file too large to process within the memory of a single machine. The unstructured nature of the data means that relational databases can be used to store and query.

Hadoop Map Reduce will be used and evaluated for this large scale data processing due to the way it partitions data across low cost software .The processed data is fed into a NoSQL database like PIG where further grouping is done to see if the surveys can be classified into clusters based on their similarity.

Headings:

- Introduction to Social Sciences Data
- Similarity calculation techniques
- Clustering
- Hadoop MapReduce
- PIG

USAGE OF LARGE SCALE DATA PROCESSING TO ANALYSE SOCIAL SCIENCES DATA

by
Ravi S Ganti

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April 2014

Approved by

_____

Table of Contents

Introduction

The dataset used for this study was a collection of survey conducted by social scientists at The Odum Institute at the University of North Carolina at Chapel Hill. These surveys ranged in topic. The motive was to find how these surveys related to one another. One way to determine these relationships was to identify commonalities between any two studies. This study aimed to employ multiple ways of defining such commonalities. Every survey has been defined in a hierarchical XML structure describing its metadata. (DDI Lite -- for DDI Codebook Version 2.0, n.d.)

Multiple metadata entities associated with a survey have been used to define the commonalities. These entities include keywords, abstract, authorized entity, time period of study, etc. For evaluating the similarity between two studies, algorithms such as cosine similarity and Term Frequency – Inverse Document Frequency (TF-IDF) have been applied against an entity (for example, keyword) from both the studies and a normalized score obtained. This score (maximum value being 1) is a measure of similarity between the two studies. The final output has all such scores for all possible pairs of studies in the dataset. A social scientist interested in the data could therefore get a feel of similarity between two surveys based on their similarity score.

One challenge in the implementation of the aforementioned algorithms was the size of the dataset, in total about 5 gigabytes. Processing the entire data on a single node machine is currently impossible. Hence Hadoop's MapReduce framework was utilized in the processing of this data. (Bhandarkar, 2010)

Similarity Techniques

Use of vector space model in document representation

Vector space modeling is used for document representation and similarity calculation in information retrieval. All the words in the collection (excluding stop words) are used to form linearly independent basis vectors. Every document is represented as a vector using each of these linearly independent basis vectors. A 1 is used if the word is present in the document and 0 if the word is not present in the document. Thus every document is represented as

$$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

Where dj is the j-th document in the collection and ($w_{1,}$, $w_{2,}$, …… $w_{n,}$) is the basis vector for the collection. Thus, vector space model converts all the documents in a collection into vectors within an n-dimensional vector space when 'n' is the number of unique words (sans stopwords) in the collection. (Arguello, n.d.)

Cosine similarity

Given the above vector space model, cosine similarity is used to calculate the similarity between two documents.

Cosine similarity between document d_1 and document d_2 = $\frac{\boldsymbol{d1}.\boldsymbol{d2}}{|\boldsymbol{d1}||\boldsymbol{d2}|}$

This value ranges between 0 and 1. The higher the value, the greater the similarity between two documents in a vector space model. (Arguello, n.d.)

Cosine similarity is binary in nature. In other words, it checks if a word is present in a document or not and gives a 1 or 0 correspondingly. This is a very useful method when the keywords of two documents are being compared as keywords usually only occur once.

Term Frequency – Inverse Document Frequency (TF.IDF)

Cosine similarity doesn't take into consideration the relative importance of words within a document. A document on Gandhi may have the word "non-violence" repeated multiple times, yet this word is of a higher importance than a word like "law" (as Gandhi had a law degree). However, the regular binary vector representation assigns both of the words with the same weight, or importance. The term frequency representation incorporates the importance of a term with respect to a document by noting the number of occurrences of the term in the document.

Likewise, the term Gandhi is of real importance in the document but occurs very rarely across the collection. The inverse document frequency (IDF), is used to accentuate this aspect. IDF favors words that do not occur in many documents.

TF and IDF in combination give a higher weight to a word that is frequent in the document but rare across the collection. The TF.IDF is an improved representation of a vector space model and is bound to improve cosine similarity if used to represent the vectors instead of the previously mentioned binary representation. (Ramos, 2003)

Eigenvectors and eigenvalues

For any square matrix A, an eigenvector is a non-zero vector v, such that A times v is equal to λ times v , where λ is a constant.

$$\boldsymbol{Av = \lambda v}$$

The value λ is called eigenvalue of vector v, corresponding to the matrix A. For a fixed value of λ and a given square matrix A, the set of vectors v which satisfy the above equations are called the eigenspace of the λ. (Elsner, 1996)

The concept of eigenspace can be extended to the creation of topic clusters for documents. For a given eigenvalue λ, we can say that documents represented by all related eigenvectors belong to the same topic. Some of the concepts which are built on eigenspaces are principal component analysis (PCA) and latent semantic analysis (LSA).

Principal component analysis (PCA)

PCA is a method that uses orthogonal transformations to convert a set of correlated variables into a set of linearly uncorrelated components. In data science, it is used to reduce the dimensionality of a data set so that the principal components retain the variation present in all of the original variables. (Jolliffe, 2005)

Ding et al. proposes a K-means clustering via PCA. K-means clustering is a common machine learning algorithm for unsupervised learning which clusters data. Ding et al.'s paper discussed how principal components can be used to classify data into clusters and form a base for K-means. The lowerbounds for K-means objective function are recalculated as total variance minus the eigenvalues of the data covariance matrix. The results find similarity between unsupervised dimension reduction use by PCA and unsupervised learning of K-means algorithm concluding that PCA can be used as a variant to calculate K-means clustering. (Ding, 1990)

Latent Semantic Analysis

Cosine similarity and TF.IDF are based on word-to-word matching of documents, although there are cases were documents are related to each other without containing the same words. The two documents may use different words to talk about similar ideas on a topic and could be conceptually close. Instead of using observed term occurrences within a document, a modified representation containing the actual similarity of a term and a document is needed. The resulting model gives us an estimate of what the observed occurrences should have been in reality. This means that if a word doesn't occur within a document, it should be present in the modified representation if it is related to the document.

Latent semantic analysis (LSA) is used to overcome this barrier and express similarity in terms of semantics rather than pairwise matching of words. LSA constructs a matrix, where the rows consist of all the words in the corpus and the columns are the documents. The individual elements in the matrix cells are the number of occurrences of a word in a document. The overlap between two matrices is used to construct a new square matrix. The element in an individual cell of this matrix is the similarity between documents represented by a row and column whose intersection is the given cell.

Singular value decomposition (SVD) is used on the above matrix to transform the individual vectors into a new space, where documents with similar semantic meaning are closer to each other. Thus, when similarity is calculated for the documents using new vectors, lack of common terms won't be of a hindrance. (Deerwester, Dumais, Furnas, Landauer, & Richard, 1990)

Hadoop MapReduce

The above algorithms could be implemented in a JAVA framework and run on a single machine if the datasets were small. However, the whole motivation behind this study was to work on large scale data processing techniques. Therefore, The Odum Institute was approached to see if they needed any large dataset to be processed. The biggest challenge of this project was to implement the same JAVA code following a MapReduce paradigm on the Hadoop framework.

The motivation behind developing the Hadoop framework is based on the knowledge that memory size is limited on a single machine and that it takes a lot of time and network bandwidth to transport data from one machine to the other. To put it in simplest terms, Hadoop framework transports "computation power to data and not data to computation." The large dataset is broken into small chunks which are stored on the Hadoop Distributed File System (HDFS) present in individual nodes. The framework has a way of optimizing memory and uses the MapReduce paradigm to reduce processing times. (Elsayed, Lin, & Oard, 2008)

MapReduce allows us to handle lists of values which are too small to fit in single memory. It has two stages called a Map and a Reduce. A Mapper class written by the user is sent to each of the nodes containing data. Mapper initially processes the data into (key,value) pairs. The Hadoop framework then writes these intermediary key, value pairs onto the Hadoop Distributed File System (HDFS). All the values corresponding to

specific key are then sent to a Reducer node, which runs the Reducer class. Thus the MapReduce programming model is suitable for computations involving large lists of key, value pairs.Word count is a classic example of a MapReduce solution to finding the number of occurrences of various words in a given corpus. It is also the basis for the code developed to process the social sciences data. A sample word count program looks as follows:

Mapper function

```
map (String key, String value):
{
        /* key – a word occurring in a particular logical input file block
           value – number of occurrences of the the word  in the logical input file block */
        for each word w in value:
        EmitIntermediate(w, 1);
}
```

The corpus, which is a large file, is broken into small individual files. Each file is sent to a particular node in the HDFS cluster. The mapper function, which is also distributed to each of the nodes, processes every logical file split and emits a key, value pair of (word, 1). As an example, for every occurrence of a word “boy,” the mapper sends a signal saying that the word “boy” has occurred once. This is done for every word in the corpus and the key, value pairs are written onto HDFS.

Reducer function

```
reduce (String key, int value):{
        /* key – a word occurrence emitted by the mapper
           value – 1 emitted by the mapper */
        int totalOccur=0;
```

```
        for each v in values:
        totalOccur++;
        Emit(AsString(totalOccur));
}
```

The number of reducers employed by Hadoop is equal to the number of keys emitted by all the mappers. Each reducer collects the (word, 1) pairs for a specific word. For example, consider the reducer which counts the number of occurrences of the word "boy" in the corpus. It collects the (boy, 1) pairs emitted by various mapper. It sums up 1 from all the pairs to find the total number of occurrences of "boy" in the entire corpus. It does the same for other words. Thus, the MapReduce programming model is suitable for quick and reliable processing of (key,value) based pairs. (Dean, 2008)

Hadoop's architecture is designed in a way that it has high fault tolerance. The main node which contains information about the locations of data across HDFS is called the Namenode. Every data block is replicated thrice across the cluster using rack-awareness replica placement policy. Two copies of a data block are stored in the same rack on HDFS and the third data block is stored in a different rack. The Namenode receives heart-beat signals from data nodes periodically. If any of the nodes fail to send a signal, Namenode replicates the data block on a new node. (Shvachko, 2010)

HDFS is built for large scale data storage and processing. It tries to make use of cheap hardware for the datanodes. The framework is also designed in such a way that horizontal expansion is easy. Some of the differences between HDFS and other distributed file systems are as follows:

- HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware.
- HDFS is designed to support write once – read many types of operations on data.
- HDFS is suitable to work on large datasets and provide high throughput access.

For running the MapReduce code, a daemon service called JobTracker, which is the main service, is run on a particular node. The JobTracker submits jobs to individual task-trackers which run near the data nodes. The JobTracker monitors individual task trackers and waits for their heartbeat signals. If a task tracker fails to respond, the Jobtracker resubmits the job to a different task tracker. Thus the JobTracker handles any failure of tasks. The hardware is usually built in such a way that the JobTracker runs on a really good hardware as it is the single point of failure. (White, 2009)

PIG Database

As mentioned before there is a need for analysis of large datasets which can be done using database products like Teradata, but they tend to cost a lot and use high level programming like SQL. The engineers who interact with this data are usually procedural in their programming style. The Map Reduce paradigm counters this by using a procedural style but too much of a procedural style makes the code expansive.

PIG Latin offers a mid-way between SQL and MapReduce by using SQL like statements for a procedural purpose. PIG runs over the Hadoop architecture reading from the HDFS and performing SQL like queries on it. PIG includes the NoSQL aspect of databases by allowing the ability to operate over plain input files that lack schema. Instead of using a single block of a SQL statement, PIG uses small multiple queries which are simpler and cleaner to use. Like MapReduce, PIG supports write once – read many type of transactions, so implementing transactional consistency and index-based lookups using schema management and pre-processing of data can be waste of resources (cons of using RDBMS). (Olston, 2008)

PIG offers nested data models which abandon first normal form (1 NF) and allows the use for sets, maps and tuples to exist as attributes in a table. This is close to the procedural style of programming natural to engineers. It allows for the use of algebraic language where every step allows a single data transformation thus allowing rich user

defined functions. PIG is easy to learn and is easy to collaborate among researchers. Development takes a lot less time compared to MapReduce. Compared to a basic code written in MapReduce, PIG also provides with a low execution time. (Gates, 2009)

Experiments

Mr.Jonathan Crabtree, Assistant Director for Information Technology and Archival Research at HW Odum Institute for Research in Social Science, gave us permission to use datasets which are of interest to the social scientists at his workplace. The datasets were loaded onto topsail-sn.unc.edu, a server in UNC. The XML structure of the individual studies in the datasets is located at the URL: http://www.ddialliance.org/sites/default/files/ddi-lite.html .

The first stage of processing the data involved parsing the individual studies. The key areas of interest were the "variable" level data (DDI Lite -- for DDI Codebook Version 2.0, n.d.) . The variable data consisted of multiple surveys per study. Each survey has a list of questions. Analysis of the questions asked in a survey can give us an idea of the surveys which in turn tells us what the study is about. Initially, a matrix was created with words as the rows, survey ids as the columns and the number of occurrences of the word in the survey as an element in a cell. A cosine similarity between any two columns would give us a measure of a similarity between the two documents represented by the columns. But as mentioned in the "Similarity Techniques" section, cosine similarity does a word-to-word match and doesn't account for semantic closeness between two documents. Thus the motivation was to cluster the surveys containing similar concepts. The output of the program (customized to MapReduce) is of the form:
(Study id, word, # of occurrences of word in a survey)

This data was fed into PIG database. Queries in PIG Latin were used to find:

1. Method 1 - Top ranked words across survey questions. Stopwords are eliminated from the words found in survey. The non-stop words are referred to as “survey words.” The GROUP BY feature was used to aggregate the survey questions by the survey words. Survey words were ranked in decreasing order of the number of the survey questions they occur in. The top 100 words were then fed into a query to find the number of studies each survey word was present in. (Please take note of the difference between occurrences in a study and occurrences in a survey)
2. Method 2 -Top ranked words across studies. Instead of ranking the survey words in the decreasing order of their occurrences in survey questions, they were ranked in the decreasing order of their occurrences in the studies. The top 100 words were then fed in a query to find the number of studies each survey word is present in.

The two modes of ranking are used to group the studies into 10 clusters. The following are the distributions after grouping the studies into clusters.

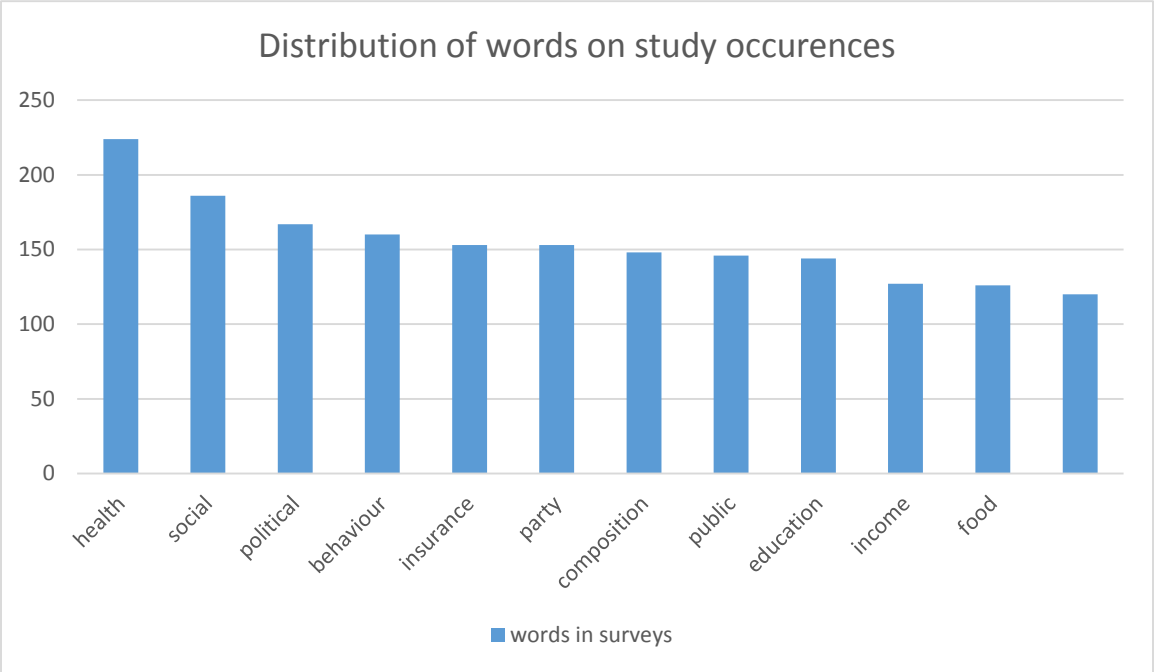

Figure1

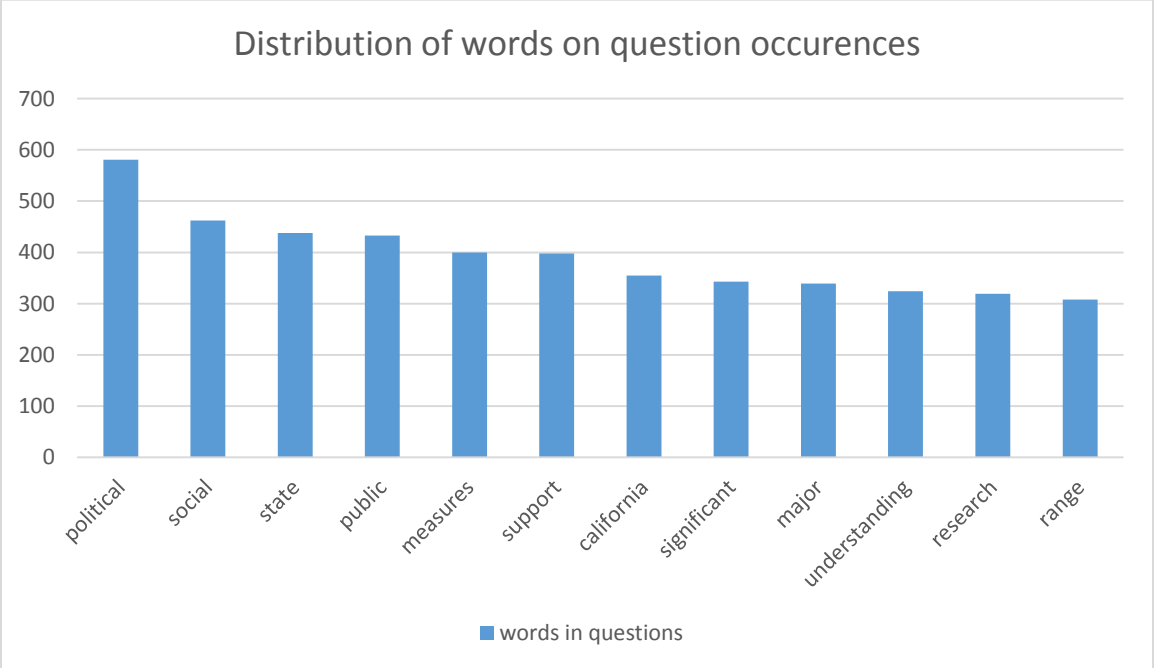

Figure 2

Figure 1 shows a balanced distribution when survey studies were clustered according to the top words found in a study. Figure 2 also gives a balances distribution when ranking is done as per individual survey questions. However, the names of the cluster in Figure 2 do not appear like comprehensive topics for a human assessor.

The next step was to find out which among the top 100 survey words were related to each other by human assessment. This was motivated by the Latent Semantic Analysis (LSA) of grouping terms which have a similar meaning .A bunch of words deemed to be related to each other where put together in a single cluster. An effort was also made to make different clusters orthogonal as used by Principal Component Analysis in creating a K-means cluster. Queries were then run to find the cluster distribution of surveys. The histograms are shown below.

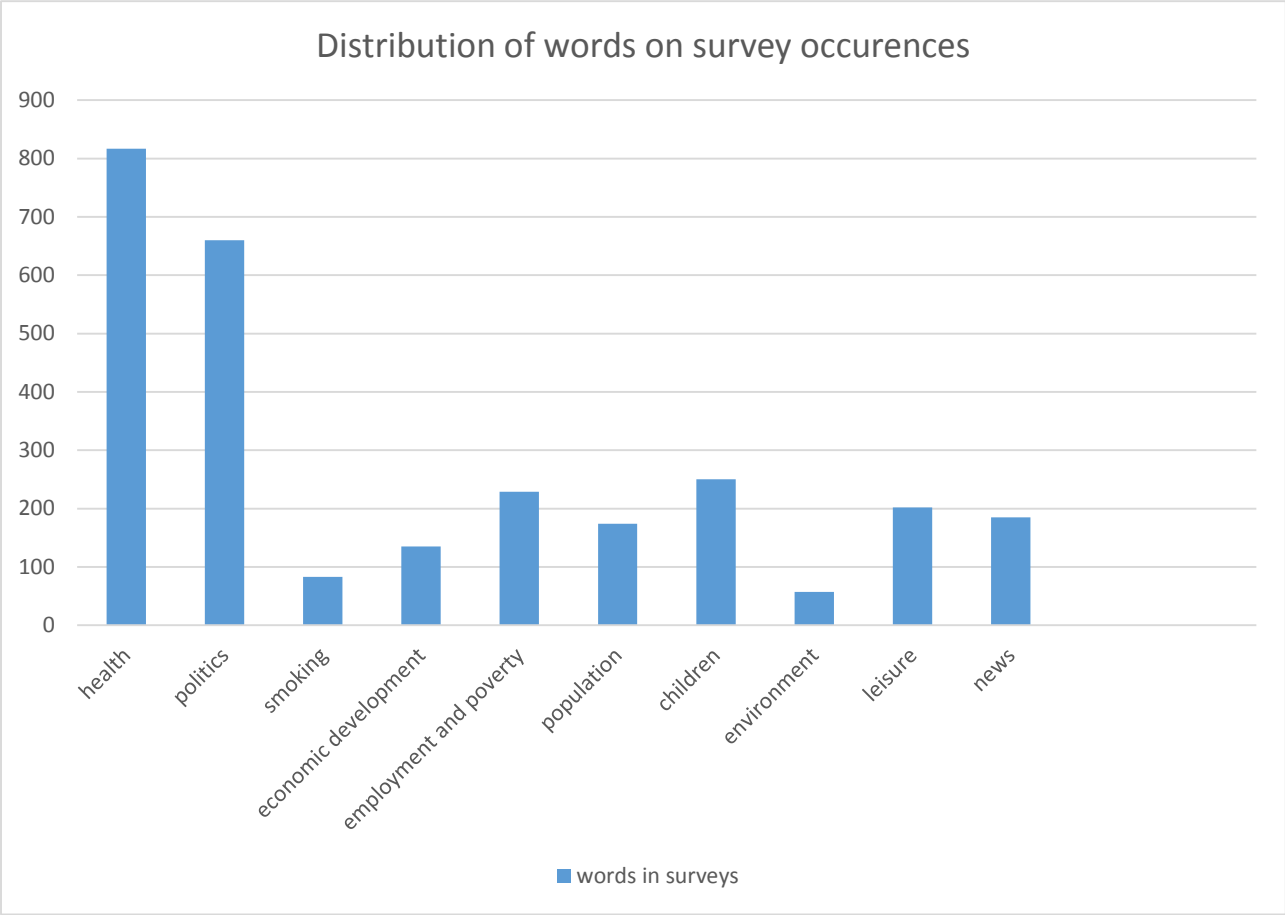
Distribution of words on survey occurences
900
800
700
600
500
400
300
200
100
0
health
politics
smoking
economic development
employment and poverty
population
children
environment
leisure
news
words in surveys

Figure 3

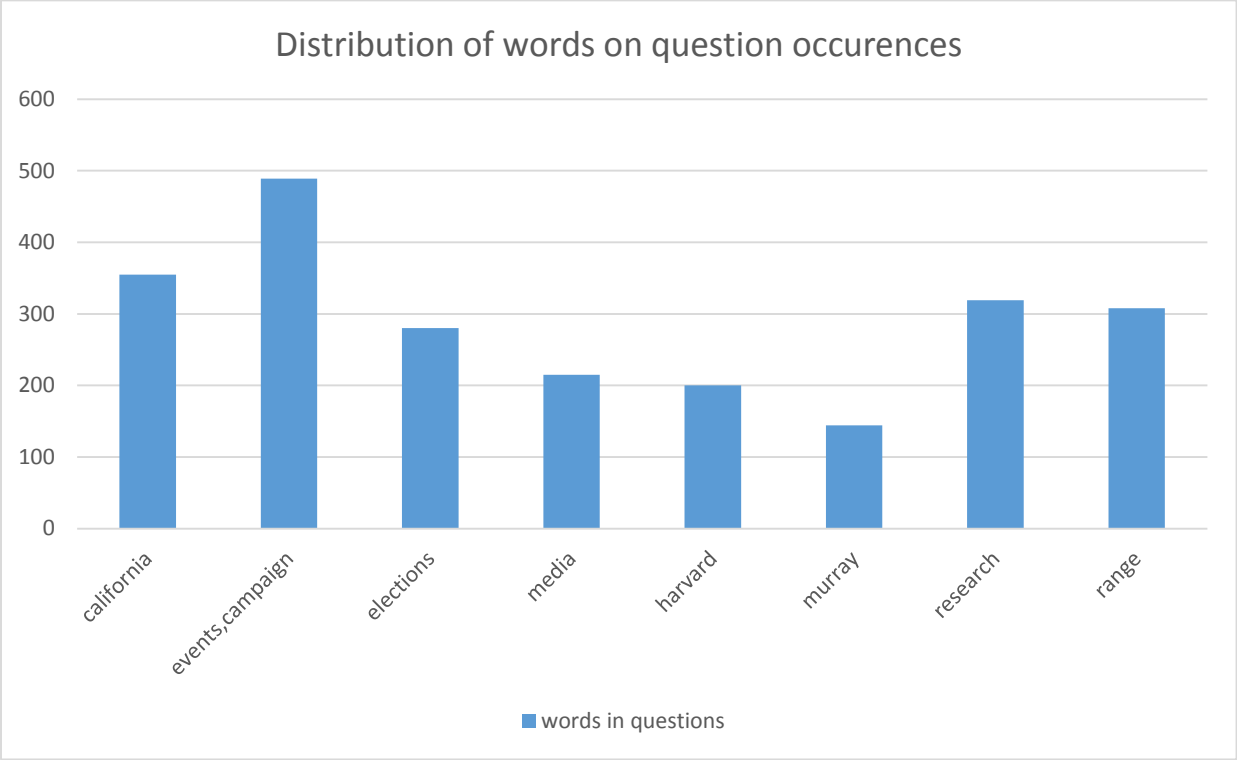

Figure 4

Figure 3 shows that most of the studies are related to health and politics which is true. The statistics retrieved from pure analysis of "survey words" in Figure 1 may not be a good indicator of topic analysis for social sciences data. An approach that considers semantics over pure keyword analysis as in Figure 3 gives a better indication of topic distribution across social sciences studies. Thus topic-wise clustering of words by a judge can make large scale data processing more accurate

Figure 4 consists of the best words in top 100. The process of clustering words prior to querying wasn't effective as the top 100 words did not really have descriptive words that could be divided into clusters. Thus method 2 was more effective than method

1 and it can be concluded that top words for a study are more useful for creating topic clusters than top words in individual survey questions.

Conclusion

XML is a widely used format to organize data in a lot of domains. When we want to compare two XML documents, we can use the techniques proposed in this study, especially when they are large datasets and need to use the MapReduce paradigm for efficient processing. The large scale data analysis and finding similarity between documents is made possible by a combination of big data technology such as Hadoop MapReduce and information retrieval techniques like cosine similarity, PCA and LSA. Hadoop is one of the most popular open source frameworks for processing large data right now.

The methodology could be applied to non XML data also, as Java parsing can be modified according to the structure of the data. Thus there is application wherever one needs to find similarity among large datasets.

Bibliography

Elsayed, T., Lin, J., & Oard, D. W. (2008). Pairwise Document Similarity in Large Collections with MapReduce. *ACM Digital Library*.

Abouzeid, A. B.-P. (2009). HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. *Proceedings of the VLDB Endowment, 2(1)*, (pp. 922-933).

Arguello, D. (n.d.). *Information Retrieval - INLS 509*. Retrieved from sils.unc.edu: http://ils.unc.edu/courses/2013_fall/inls509_001/lectures/06-VectorSpaceModel.pdf

Bhandarkar, M. (2010). MapReduce Programming with Apache Hadoop. *IPDPS 2010 Symposium on Parallel & Distributed Processing (IPDPS).* Atlanta , GA.

Chu, C. T. (2006). Map-reduce for machine learning on multicore. *NIPS (Vol. 6)*, 281-288.

DDI Lite -- for DDI Codebook Version 2.0. (n.d.). Retrieved from Data Documentation Initiative: http://www.ddialliance.org/sites/default/files/ddi-lite.html

Dean, J. &. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM, 51(1)*, (pp. 107-113).

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Richard. (1990). Indexing by Latent Semantic Analysis. *American Societ for Information Science*, 391.

Ding, C. &. (n.d.). K-means clustering via principal component analysis. *In Proceedings of the twenty-first international conference on Machine learning* (p. 29). ACM.

Elsner, J. B. (1996). Eigenvalues and Eigenvectors. Singular Spectrum Analysis.

Gates, A. F. (2009). Building a high-level dataflow system on top of Map-Reduce: the Pig experience. *Proceedings of the VLDB Endowment,,* (pp. 1414-1425).

Jolliffe, I. (2005). *Principal component analysis.* John Wiley & Sons, Ltd.

Olston, C. R. (2008). Pig latin: a not-so-foreign language for data processing. . *In Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (pp. 1099-1110). ACM.

Ramos, J. (2003). Using TF-IDF to Determine Word Relevance in Document Queries. *Proceedings of the First Instructional Conference on Machine Learning.* Retrieved from Computer Science - Rutgers University.

Shvachko, K. K. (2010). The hadoop distributed file system. *In Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium*, 1-10.

White, T. (2009). *Hadoop: The Definitive Guide.* O'Reilly Media.

Appendix

MapReduce code for parsing XML document

```
package com.org;
//import javax.xml.stream.XMLStreamConstants;//XMLInputFactory;
import java.io.*;
import org.apache.hadoop.*;
import javax.xml.stream.*;
import java.util.*;
public class OdumParser
{

	public static class XmlInputFormat1 extends TextInputFormat
		{

		public static final String START_TAG_KEY = "xmlinput.start";
		public static final String END_TAG_KEY = "xmlinput.end";


		public RecordReader<LongWritable, Text> createRecordReader(InputSplit split, TaskAttemptContext
context)
			{
			return new XmlRecordReader();
		}

	/**
	 * XMLRecordReader class to read through a given xml document to output
	 * xml blocks as records as specified by the start tag and end tag
	 *
	 */
	// @Override
		public static class XmlRecordReader extends RecordReader<LongWritable, Text>
			{
			private byte[] startTag;
			private byte[] endTag;
			private long start;
			private long end;
			private FSDataInputStream fsin;
			private DataOutputBuffer buffer = new DataOutputBuffer();

			private LongWritable key = new LongWritable();
			private Text value = new Text();
			@Override
			public void initialize(InputSplit split, TaskAttemptContext context)
			throws IOException, InterruptedException
				{
				Configuration conf = context.getConfiguration();
				startTag = conf.get(START_TAG_KEY).getBytes("utf-8");
				endTag = conf.get(END_TAG_KEY).getBytes("utf-8");
				FileSplit fileSplit = (FileSplit) split;
```

```
            // open the file and seek to the start of the split

            start = fileSplit.getStart();
            end = start + fileSplit.getLength();
            Path file = fileSplit.getPath();
            FileSystem fs = file.getFileSystem(conf);
            fsin = fs.open(fileSplit.getPath());
            fsin.seek(start);

}
@Override
public boolean nextKeyValue() throws IOException,
InterruptedException
            {
            if (fsin.getPos() < end)
                        {
                        if (readUntilMatch(startTag, false))
                                    {
                        try
                                                {
                                    buffer.write(startTag);
                                    if (readUntilMatch(endTag, true))
                                                            {
                                                key.set(fsin.getPos());
                                                value.set(buffer.getData(), 0,
                                    buffer.getLength());
                                                return true;
                                    }
                        } finally
                                                {
                                    buffer.reset();
                        }
                        }
            }
            return false;
}
@Override
public LongWritable getCurrentKey() throws IOException,
InterruptedException
            {
            return key;
}

            @Override
public Text getCurrentValue() throws IOException,
InterruptedException
            {
            return value;
}
@Override
public void close() throws IOException
            {
            fsin.close();
}
@Override
public float getProgress() throws IOException
            {
            return (fsin.getPos() - start) / (float) (end - start);
}
```

```
                private boolean readUntilMatch(byte[] match, boolean withinBlock)
                throws IOException
                        {
                        int i = 0;
                        while (true)
                                {
                                int b = fsin.read();
                                // end of file:
                                if (b == -1)
                        return false;
                                // save to buffer:
                                if (withinBlock)
                                 buffer.write(b);
                                // check if we're matching:
                                if (b == match[i])
                                        {
                                i++;
                                if (i >= match.length)
                                return true;
                                }
                                        else
                        i = 0;
                                // see if we've passed the stop point:

                                        if (!withinBlock && i == 0 && fsin.getPos() >= end)
                        return false;
                        }
                }


        }


        }


        public static class Map extends Mapper<LongWritable, Text,
        Text, Text>
        {
                @Override
                protected void map(LongWritable key, Text value,
            Mapper.Context context)throws IOException, InterruptedException
                 {
                        String document = value.toString();
                        // System.out.println("""" + document + """");
                try
                         {
                                XMLStreamReader reader =
XMLInputFactory.newInstance().createXMLStreamReader(new ByteArrayInputStream(document.getBytes()));
                                 String propertyName = "";
                                String propertyValue = "";
                                String currentElement = "";
                                String pageTitle = "";
                                String pageText = "";
                                while (reader.hasNext())
                                 {
                                int code = reader.next();
                                switch (code)
                                        {
```

```
                                        case XMLStreamConstants.START_ELEMENT:
//START_ELEMENT:
                                        currentElement = reader.getLocalName();
                                         break;
                                        case XMLStreamConstants.CHARACTERS:  //CHARACTERS:
                                        if (currentElement.equalsIgnoreCase("idno agency"))
                                                {
                                                propertyName += reader.getText();
                                                //System.out.println(propertyName);
                                        }
                                                else if (currentElement.equalsIgnoreCase("labl
level=\"variable\""))
                                                 {
                                                propertyValue += reader.getText();
                                                //System.out.println(propertyValue);
                                        }
                                        break;
                                }
                                }
                                reader.close();
                                pageTitle = propertyName.replaceAll("[()?:!;~`@#$%^&*={}|<>]+","");
                                pageText = propertyValue.replaceAll("[()?:!;~`@#$%^&*={}|<>]+","");
                                context.write(new Text(pageTitle.trim()), new Text(pageText.trim()));

                        }
                catch(XMLStreamException e)
                        {
                        // context.getCounter(INVALID_RECORDS).increment(1);

                }
                }
        }
        public static class Reduce
        extends Reducer<Text, Text, Text, Text>
         {

                @Override
                protected void setup(Context context)
                throws IOException, InterruptedException
                 {
                        //context.write(new Text("<configuration>"), null);
                }

                @Override
                protected void cleanup(Context context)
                throws IOException, InterruptedException
                 {
                        //context.write(new Text("</configuration>"), null);
                }

                private Text outputKey = new Text();
                 public void reduce(Text key, Iterable<Text> values,
           Context context)throws IOException, InterruptedException
                 {
                        for (Text value : values)
                         {
                                outputKey.set(constructPropertyXml(key, value));
                                context.write(outputKey, null);
                        }
                }
```

```
		public static String constructPropertyXml(Text name, Text value)
		{
			java.util.Map  D = new HashMap();
			//String name1 = name.toString();
			//String name2 = name1.replaceAll("[()?:!.,;]+","");
			//		HashMapWritable<String , Integer> D = new HashMapWritable<String ,
Integer>();
			String body = value.toString();

			StringTokenizer tokenizer = new StringTokenizer(body);
		while (tokenizer.hasMoreTokens())
			{
				String temp = tokenizer.nextToken();
				Integer count = (Integer)D.get(temp);
				if(count == null)
				{
					count = 0;
				}
				D.put(temp, count + 1);

			}

			D.keySet().removeAll(Arrays.asList(stopWords));

			StringBuilder sb = new StringBuilder();

			Iterator iter = D.keySet().iterator();
			while(iter.hasNext())
			{

				String key=(String)iter.next();
				//String key1= key.replaceAll("[()?:!.,;]+","");
				Integer val=(Integer)D.get(key);
				sb.append(name).append(":").append(key).append(":")
			.append(val).append("\n");
			}
			return sb.toString();

		}
	}

    public static void main(String[] args) throws Exception
    {
        Configuration conf = new Configuration();

        conf.set("xmlinput.start", "<codeBook>");
        conf.set("xmlinput.end", "</codeBook>");

                conf.setBoolean("mapred.compress.map.output", true);
  conf.setClass("mapred.map.output.compression.codec", org.apache.hadoop.io.compress.GzipCodec.class,
    org.apache.hadoop.io.compress.CompressionCodec.class);

        Job job = new Job(conf);  //configure the job, submit it, control its execution, and query the state
        job.setJarByClass(OdumParser.class); //set jar by finding where the class came from
        job.setOutputKeyClass(Text.class); //Set the key class for the job output data
        job.setOutputValueClass(Text.class);

        //job.setCompressMapOutput(true);
                //job.setMapOutputCompressorClass(GzipCodec.class);

                //job.setCompressOutput(job, true);
```

```
                //job.setClass("mapred.output.compression.codec", GzipCodec.class,CompressionCodec.class);
                job.setMapperClass(OdumParser.Map.class);
        job.setReducerClass(OdumParser.Reduce.class);

        job.setInputFormatClass(XmlInputFormat1.class);  //Set the InputFormat for the job
        job.setOutputFormatClass(TextOutputFormat.class); //Set the OutputFormat for the job

                FileOutputFormat.setCompressOutput(job,true);

        FileOutputFormat.setOutputCompressorClass(job,org.apache.hadoop.io.compress.GzipCodec.class);

        FileInputFormat.addInputPath(job, new Path(args[0])); //the job for which the input path should be modified
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
    }
}
```

PIG code

```
studies1 = LOAD 'study.txt' USING PigStorage(',') as (id, key, count:double);
A = GROUP studies1 by key;
B = FOREACH A GENERATE group,COUNT(studies1);
C = ORDER B by $1 DESC;
D = LIMIT C 100;
DUMP D;

studies1 = LOAD 'survey.txt' USING PigStorage(',') as (id, key, count:double);
A = GROUP studies1 by key;
B = FOREACH A GENERATE group,COUNT(studies1);
C = ORDER B by $1 DESC;
D = LIMIT C 100;
DUMP D;
```

DTD for XML data

Highlighted/bold elements comprise the DDI Lite, a list of recommended elements of the full DTD. I.e., those are the fields you really need to make your XML useable.

DDI Lite -- for DDI Codebook Version 2.0

```
* == ELEMENT IS OPTIONAL & REPEATABLE
+ == ELEMENT IS MANDATORY & REPEATABLE
? == ELEMENT IS OPTIONAL & NON-REPEATABLE
  == ELEMENT IS MANDATORY & NON-REPEATABLE

0.0 codeBook (ATT == ID, xml-lang, source, version)
|
|---- 1.0 docDscr* (ATT == ID, xml-lang, source)
|     |
|     |---- 1.1 citation? (ATT == ID, xml-lang, source, MARCURI)
|     |     |
|     |     |---- 1.1.1 titlStmt (ATT == ID, xml-lang, source)
|     |     |     |
|     |     |     |---- 1.1.1.1 titl     (ATT == ID, xml-lang, source)
```

```
|   |   |   |---- 1.1.1.2 subTitl* (ATT == ID, xml-lang, source)
|   |   |   |---- 1.1.1.3 altTitl* (ATT == ID, xml-lang, source)
|   |   |   |---- 1.1.1.4 parTitl* (ATT == ID, xml-lang, source)
|   |   |   +---- 1.1.1.5 IDNo*    (ATT == ID, xml-lang, source, agency, level)
|   |   |
|   |   |---- 1.1.2 rspStmt? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 1.1.2.1 AuthEnty*      (ATT == ID, xml-lang, source, affiliation)
|   |   |   +---- 1.1.2.2 othId*        (ATT == ID, xml-lang, source, type, role,
|   |   |                                affiliation)
|   |   |
|   |   |---- 1.1.3 prodStmt? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 1.1.3.1 producer*     (ATT == ID, xml-lang, source,abbr, affiliation,
|   |   |   |                            role)
|   |   |   |---- 1.1.3.2 copyright?    (ATT == ID, xml-lang, source)
|   |   |   |---- 1.1.3.3 prodDate*     (ATT == ID, xml-lang, source, date)
|   |   |   |---- 1.1.3.4 prodPlac*     (ATT == ID, xml-lang, source)
|   |   |   |---- 1.1.3.5 software*     (ATT == ID, xml-lang, source, date, version)
|   |   |   |---- 1.1.3.6 fundAg*       (ATT == ID, xml-lang, source, abbr, role)
|   |   |   +---- 1.1.3.7 grantNo*      (ATT == ID, xml-lang, source, agency, role)
|   |   |
|   |   |---- 1.1.4 distStmt? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 1.1.4.1 distrbtr* (ATT == ID, xml-lang, source, abbr, affiliation, URI)
|   |   |   |---- 1.1.4.2 contact*  (ATT == ID, xml-lang, source, affiliation, URI, email)
|   |   |   |---- 1.1.4.3 depositr* (ATT == ID, xml-lang, source, abbr, affiliation)
|   |   |   |---- 1.1.4.4 depDate*  (ATT == ID, xml-lang, source, date)
|   |   |   +---- 1.1.4.5 distDate? (ATT == ID, xml-lang, source, date)
|   |   |
|   |   |---- 1.1.5 serStmt? (ATT == ID, xml-lang, source, URI)
|   |   |   |
|   |   |   |---- 1.1.5.1 serName* (ATT == ID, xml-lang, source, abbr)
|   |   |   +---- 1.1.5.2 serInfo* (ATT == ID, xml-lang, source)
|   |   |
|   |   |---- 1.1.6 verStmt* (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 1.1.6.1 version? (ATT == ID, xml-lang, source, type, date)
|   |   |   |---- 1.1.6.2 verResp? (ATT == ID, xml-lang, source, affiliation)
|   |   |   +---- 1.1.6.3 notes*   (ATT == ID, xml-lang, source, type, subject, level, resp,
|   |   |                           sdatrefs)
|   |   |
|   |   |---- 1.1.7 biblCit?(ATT == ID, xml-lang, source, format)
|   |   |---- 1.1.8 holdings* (ATT == ID, xml-lang, source, location, callno, URI, media)
|   |   +---- 1.1.9 notes*    (ATT == ID, xml-lang, source, type, subject, level, resp,
|   |                          sdatrefs)
|   |
|   |---- 1.2 guide?     (ATT == ID, xml-lang, source)
|   |---- 1.3 docStatus? (ATT == ID, xml-lang, source)
|   |---- 1.4 docSrc*    (ATT == ID, xml-lang, source, MARCURI)
|   |   |
|   |   |---- 1.4.1 titlStmt (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 1.4.1.1 titl     (ATT == ID, xml-lang, source)
|   |   |   |---- 1.4.1.2 subTitl* (ATT == ID, xml-lang, source)
|   |   |   |---- 1.4.1.3 altTitl* (ATT == ID, xml-lang, source)
|   |   |   |---- 1.4.1.4 parTitl* (ATT == ID, xml-lang, source)
|   |   |   +---- 1.4.1.5 IDNo*    (ATT == ID, xml-lang, source, agency, level)
|   |   |
|   |   |---- 1.4.2 rspStmt? (ATT == ID, xml-lang, source)
|   |   |   |
```

```
|   |   |   |---- 1.4.2.1 AuthEnty*     (ATT == ID, xml-lang, source, affiliation)
|   |   |   +---- 1.4.2.2 othId*        (ATT == ID, xml-lang, source, type, role, affiliation)
|   |   |
|   |   |---- 1.4.3 prodStmt? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 1.4.3.1 producer*    (ATT == ID, xml-lang, source,abbr, affiliation, role)
|   |   |   |---- 1.4.3.2 copyright?   (ATT == ID, xml-lang, source)
|   |   |   |---- 1.4.3.3 prodDate*    (ATT == ID, xml-lang, source, date)
|   |   |   |---- 1.4.3.4 prodPlac*    (ATT == ID, xml-lang, source)
|   |   |   |---- 1.4.3.5 software*    (ATT == ID, xml-lang, source, date, version)
|   |   |   |---- 1.4.3.6 fundAg*      (ATT == ID, xml-lang, source, abbr, role)
|   |   |   +---- 1.4.3.7 grantNo*     (ATT == ID, xml-lang, source, agency, role)
|   |   |
|   |   |---- 1.4.4 distStmt? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 1.4.4.1 distrbtr* (ATT == ID, xml-lang, source, abbr, affiliation, URI)
|   |   |   |---- 1.4.4.2 contact*(ATT == ID, xml-lang, source, affiliation, URI, email)
|   |   |   |---- 1.4.4.3 depositr* (ATT == ID, xml-lang, source, abbr, affiliation)
|   |   |   |---- 1.4.4.4 depDate*(ATT == ID, xml-lang, source, date)
|   |   |   +---- 1.4.4.5 distDate? (ATT == ID, xml-lang, source, date)
|   |   |
|   |   |---- 1.4.5 serStmt? (ATT == ID, xml-lang, source, URI)
|   |   |   |
|   |   |   |---- 1.4.5.1 serName* (ATT == ID, xml-lang, source, abbr)
|   |   |   +---- 1.4.5.2 serInfo* (ATT == ID, xml-lang, source)
|   |   |
|   |   |---- 1.4.6 verStmt* (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 1.4.6.1 version? (ATT == ID, xml-lang, source, type, date)
|   |   |   |---- 1.4.6.2 verResp? (ATT == ID, xml-lang, source, affiliation)
|   |   |   +---- 1.4.6.3 notes*   (ATT == ID, xml-lang, source, type, subject, level, resp,
|   |   |                           sdatrefs)
|   |   |
|   |   |---- 1.4.7 biblCit?(ATT == ID, xml-lang, source, format)
|   |   |---- 1.4.8 holdings* (ATT == ID, xml-lang, source, location, callno, URI, media)
|   |   +---- 1.4.9 notes*    (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|   |
|   +---- 1.5 notes*     (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|
|---- 2.0 stdyDscr+ (ATT == ID, xml-lang, source, access)
|   |
|   |---- 2.1 citation+ (ATT == ID, xml-lang, source, MARCURI)
|   |   |
|   |   |---- 2.1.1 titlStmt (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.1.1.1 titl     (ATT == ID, xml-lang, source)
|   |   |   |---- 2.1.1.2 subTitl* (ATT == ID, xml-lang, source)
|   |   |   |---- 2.1.1.3 altTitl* (ATT == ID, xml-lang, source)
|   |   |   |---- 2.1.1.4 parTitl* (ATT == ID, xml-lang, source)
|   |   |   +---- 2.1.1.5 IDNo*    (ATT == ID, xml-lang, source, agency, level)
|   |   |
|   |   |---- 2.1.2 rspStmt? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.1.2.1 AuthEnty*     (ATT == ID, xml-lang, source, affiliation)
|   |   |   +---- 2.1.2.2 othId*         (ATT == ID, xml-lang, source, type, role, affiliation)
|   |   |
|   |   |---- 2.1.3 prodStmt? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.1.3.1 producer*     (ATT == ID, xml-lang, source, abbr, affilation, role)
|   |   |   |---- 2.1.3.2 copyright?    (ATT == ID, xml-lang, source)
|   |   |   |---- 2.1.3.3 prodDate*     (ATT == ID, xml-lang, source, date)
```

```
|   |   |   |---- 2.1.3.4 prodPlac*     (ATT == ID, xml-lang, source)
|   |   |   |---- 2.1.3.5 software*     (ATT == ID, xml-lang, source, date, version)
|   |   |   |---- 2.1.3.6 fundAg*       (ATT == ID, xml-lang, source, abbr, role)
|   |   |   +---- 2.1.3.7 grantNo*      (ATT == ID, xml-lang, source, agency, role)
|   |   |
|   |   |---- 2.1.4 distStmt? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.1.4.1 distrbtr* (ATT == ID, xml-lang, source, abbr, affiliation, URI)
|   |   |   |---- 2.1.4.2 contact*  (ATT == ID, xml-lang, source, affiliation, URI, email)
|   |   |   |---- 2.1.4.3 depositr* (ATT == ID, xml-lang, source, abbr, affiliation)
|   |   |   |---- 2.1.4.4 depDate*  (ATT == ID, xml-lang, source, date)
|   |   |   +---- 2.1.4.5 distDate? (ATT == ID, xml-lang, source, date)
|   |   |
|   |   |---- 2.1.5 serStmt? (ATT == ID, xml-lang, source, URI)
|   |   |   |
|   |   |   |---- 2.1.5.1 serName* (ATT == ID, xml-lang, source, abbr)
|   |   |   +---- 2.1.5.2 serInfo* (ATT == ID, xml-lang, source)
|   |   |
|   |   |---- 2.1.6 verStmt* (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.1.6.1 version? (ATT == ID, xml-lang, source, type, date)
|   |   |   |---- 2.1.6.2 verResp? (ATT == ID, xml-lang, source, affiliation)
|   |   |   +---- 2.1.6.3 notes*   (ATT == ID, xml-lang, source, type, subject, level, resp,
|   |   |                           sdatrefs)
|   |   |
|   |   |---- 2.1.7 biblCit?  (ATT == ID, xml-lang, source, format)
|   |   |---- 2.1.8 holdings* (ATT == ID, xml-lang, source, location, callno, URI, media)
|   |   +---- 2.1.9 notes*    (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|   |
|   |---- 2.2 stdyInfo* (ATT == ID, xml-lang, source)
|   |   |
|   |   |---- 2.2.1 subject? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.2.1.1 keyword*  (ATT == ID, xml-lang, source, vocab, vocabURI)
|   |   |   +---- 2.2.1.2 topcClas* (ATT == ID, xml-lang, source, vocab, vocabURI)
|   |   |
|   |   |---- 2.2.2 abstract* (ATT == ID, xml-lang, source, date)
|   |   |
|   |   |---- 2.2.3 sumDscr* (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.2.3.1 timePrd*   (ATT == ID, xml-lang, source, event, date, cycle)
|   |   |   |---- 2.2.3.2 collDate*  (ATT == ID, xml-lang, source, event, date, cycle)
|   |   |   |---- 2.2.3.3 nation*    (ATT == ID, xml-lang, source, abbr)
|   |   |   |---- 2.2.3.4 geogCover* (ATT == ID, xml-lang, source)
|   |   |   |---- 2.2.3.5 geogUnit*  (ATT == ID, xml-lang, source)
|   |   |   |---- 2.2.3.6 geoBndBox? (ATT == ID, xml-lang, source)
|   |   |   |   |
|   |   |   |   |---- 2.2.3.6.1 westBL (ATT == ID, xml-lang, source)
|   |   |   |   |---- 2.2.3.6.2 eastBL (ATT == ID, xml-lang, source)
|   |   |   |   |---- 2.2.3.6.3 southBL (ATT == ID, xml-lang, source)
|   |   |   |   +---- 2.2.3.6.4 northBL (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.2.3.7 boundPoly? (ATT == ID, xml-lang, source)
|   |   |   |   |
|   |   |   |   +---- 2.2.3.7.1 polygon+ (ATT == ID, xml-lang, source)
|   |   |   |       |
|   |   |   |       +---- 2.2.3.7.1.1 point+  (ATT == ID, xml-lang, source)
|   |   |   |           |
|   |   |   |           |---- 2.2.3.7.1.1.1 gringLat (ATT == ID, xml-lang, source)
|   |   |   |           +---- 2.2.3.7.1.1.2 gringLon (ATT == ID, xml-lang, source)
|   |   |   |
```

```
|   |   |   |---- 2.2.3.8 anlyUnit*  (ATT == ID, xml-lang, source, unit)
|   |   |   |---- 2.2.3.9 universe*  (ATT == ID, xml-lang, source, level, clusion)
|   |   |   +---- 2.2.3.10 dataKind*  (ATT == ID, xml-lang, source)
|   |   |
|   |   +---- 2.2.4 notes* (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|   |
|   |---- 2.3 method* (ATT == ID, xml-lang, source)
|   |   |
|   |   |---- 2.3.1 dataColl* (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.3.1.1 timeMeth*     (ATT == ID, xml-lang, source, method)
|   |   |   |---- 2.3.1.2 dataCollector* (ATT == ID, xml-lang, source, abbr, affiliation)
|   |   |   |---- 2.3.1.3 frequenc*     (ATT == ID, xml-lang, source, freq)
|   |   |   |---- 2.3.1.4 sampProc*     (ATT == ID, xml-lang, source)
|   |   |   |---- 2.3.1.5 deviat*       (ATT == ID, xml-lang, source)
|   |   |   |---- 2.3.1.6 collMode*     (ATT == ID, xml-lang, source)
|   |   |   |---- 2.3.1.7 resInstru*    (ATT == ID, xml-lang, source, type)
|   |   |   |---- 2.3.1.8 sources?      (ATT == ID, xml-lang, source)<-----+
|   |   |   |   |                                                        |
|   |   |   |   |---- 2.3.1.8.1 dataSrc* (ATT == ID, xml-lang, source)   |
|   |   |   |   |---- 2.3.1.8.2 srcOrig* (ATT == ID, xml-lang, source)   |
|   |   |   |   |---- 2.3.1.8.3 srcChar* (ATT == ID, xml-lang, source)   |
|   |   |   |   |---- 2.3.1.8.4 srcDocu* (ATT == ID, xml-lang, source)   |
|   |   |   |   +---- 2.3.1.8.5 sources* (ATT == ID, xml-lang, source) ---+
|   |   |   |   NOTE: ELEMENT sources has recursive definition,
|   |   |   |        so anywithin a codebook can
|   |   |   |        themselves list multiple, subsidiary sources.
|   |   |   |
|   |   |   |---- 2.3.1.9 collSitu*     (ATT == ID, xml-lang, source)
|   |   |   |---- 2.3.1.10 actMin*      (ATT == ID, xml-lang, source)
|   |   |   |---- 2.3.1.11 ConOps*      (ATT == ID, xml-lang, source, agency)
|   |   |   |---- 2.3.1.12 weight*      (ATT == ID, xml-lang, source)
|   |   |   +---- 2.3.1.13 cleanOps*    (ATT == ID, xml-lang, source, agency)
|   |   |
|   |   |---- 2.3.2 notes*    (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|   |   |---- 2.3.3 anlyInfo? (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.3.3.1 respRate*  (ATT == ID, xml-lang, source)
|   |   |   |---- 2.3.3.2 EstSmpErr* (ATT == ID, xml-lang, source)
|   |   |   +---- 2.3.3.3 dataAppr*  (ATT == ID, xml-lang, source)
|   |   |
|   |   +---- 2.3.4 stdyClas? (ATT == ID, xml-lang, source, type)
|   |
|   |---- 2.4 dataAccs* (ATT == ID, xml-lang, source)
|   |   |
|   |   |---- 2.4.1 setAvail* (ATT == ID, xml-lang, source, media, callno, label, type)
|   |   |   |
|   |   |   |---- 2.4.1.1 accsPlac*  (ATT == ID, xml-lang, source, URI)
|   |   |   |---- 2.4.1.2 origArch?  (ATT == ID, xml-lang, source)
|   |   |   |---- 2.4.1.3 avlStatus* (ATT == ID, xml-lang, source)
|   |   |   |---- 2.4.1.4 collSize?  (ATT == ID, xml-lang, source)
|   |   |   |---- 2.4.1.5 complete?  (ATT == ID, xml-lang, source)
|   |   |   |---- 2.4.1.6 fileQnty?  (ATT == ID, xml-lang, source)
|   |   |   +---- 2.4.1.7 notes*     (ATT == ID, xml-lang, source, type, subject, level,
|   |   |                             resp, sdatrefs)
|   |   |
|   |   |-----2.4.2 useStmt* (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 2.4.2.1 confDec?    (ATT == ID, xml-lang, source, required, formNo, URI)
|   |   |   |---- 2.4.2.2 specPerm?   (ATT == ID, xml-lang, source, required, formNo, URI)
|   |   |   |---- 2.4.2.3 restrctn?   (ATT == ID, xml-lang, source)
```

```
|    |    |    |---- 2.4.2.4 contact*    (ATT == ID, xml-lang, source, affiliation, URI, email)
|    |    |    |---- 2.4.2.5 citReq?     (ATT == ID, xml-lang, source)
|    |    |    |---- 2.4.2.6 deposReq?   (ATT == ID, xml-lang, source)
|    |    |    |---- 2.4.2.7 conditions? (ATT == ID, xml-lang, source)
|    |    |    +---- 2.4.2.8 disclaimer? (ATT == ID, xml-lang, source)
|    |    |
|    |    +---- 2.4.3 notes* (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|    |
|    |---- 2.5 othrStdyMat* (ATT == ID, xml-lang, source)
|    |    |
|    |    |---- 2.5.1 relMat*(ATT == ID, xml-lang, source, callno, label, type, media)
|    |    |    |
|    |    |    +---- 2.5.1.1 citation* (ATT == ID, xml-lang, source, MARCURI)
|    |    |
|    |    |---- 2.5.2 relStdy* (ATT == ID, xml-lang, source)
|    |    |    |
|    |    |    +---- 2.5.2.1 citation* (ATT == ID, xml-lang, source, MARCURI)
|    |    |
|    |    |---- 2.5.3 relPubl* (ATT == ID, xml-lang, source)
|    |    |    |
|    |    |    +---- 2.5.3.1 citation* (ATT == ID, xml-lang, source, MARCURI)
|    |    |
|    |    +---- 2.5.4 othRefs* (ATT == ID, xml-lang, source)
|    |         |
|    |         +---- 2.5.4.1 citation* (ATT == ID, xml-lang, source, MARCURI)
|    |                   NOTE: complete tree under citation element omitted for
|    |                   reasons of space for 2.5.1.1, 2.5.2.1, 2.5.3.1, and 2.5.4.1
|    |
|    +---- 2.6 notes* (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|
|
|---- 3.0 fileDscr* (ATT == ID, xml-lang, source, URI, sdatrefs, methrefs, pubrefs, access)
|    |
|    |---- 3.1 fileTxt* (ATT == ID, xml-lang, source)
|    |    |
|    |    |---- 3.1.1 fileName? (ATT ==ID, xml-lang, source)
|    |    |---- 3.1.2 fileCont? (ATT == ID, xml-lang, source)
|    |    |---- 3.1.3 fileStrc? (ATT == ID, xml-lang, source, type)
|    |    |    |
|    |    |    |---- 3.1.3.1 recGrp* (ATT == ID, xml-lang, source, recGrp, rectype, keyvar,
|    |    |    |    |                    recidvar, rtypeloc, rtypewidth, rtypevtype)
|    |    |    |    |
|    |    |    |    |---- 3.1.3.1.1 labl*      (ATT == ID, xml-lang, source, level, vendor, country, sdatrefs)
|    |    |    |    +---- 3.1.3.1.2 recDimnsn? (ATT == ID, xml-lang, source, level)
|    |    |    |         |
|    |    |    |         |---- 3.1.3.1.2.1 varQnty?  (ATT == ID, xml-lang, source)
|    |    |    |         |---- 3.1.3.1.2.2 caseQnty? (ATT == ID, xml-lang, source)
|    |    |    |         +---- 3.1.3.1.2.3 logRecL?  (ATT == ID, xml-lang, source)
|    |    |    |
|    |    |    +---- 3.1.3.2 notes* (ATT == ID, xml-lang, source, type, subject, level, resp,
|    |    |                          sdatrefs)
|    |    |
|    |    |---- 3.1.4 dimensns? (ATT == ID, xml-lang, source)
|    |    |    |
|    |    |    |---- 3.1.4.1 caseQnty*  (ATT == ID, xml-lang, source)
|    |    |    |---- 3.1.4.2 varQnty*   (ATT == ID, xml-lang, source)
|    |    |    |---- 3.1.4.3 logRecL*   (ATT == ID, xml-lang, source)
|    |    |    |---- 3.1.4.4 recPrCas*  (ATT == ID, xml-lang, source)
|    |    |    +---- 3.1.4.5 recNumTot* (ATT == ID, xml-lang, source)
|    |    |
|    |    |---- 3.1.5 fileType?  (ATT == ID, xml-lang, source, charset)
```

```
|    |    |---- 3.1.6 format?    (ATT == ID, xml-lang, source)
|    |    |---- 3.1.7 filePlac?  (ATT == ID, xml-lang, source)
|    |    |---- 3.1.8 dataChck*  (ATT == ID, xml-lang, source)
|    |    |---- 3.1.9 ProcStat?  (ATT == ID, xml-lang, source)
|    |    |---- 3.1.10 dataMsng? (ATT == ID, xml-lang, source)
|    |    |---- 3.1.11 software* (ATT == ID, xml-lang, source, date, version)
|    |    +---- 3.1.12 verStmt?  (ATT == ID, xml-lang, source)
|    |         |
|    |         |---- 3.1.12.1 version? (ATT == ID, xml-lang, source, type, date)
|    |         |---- 3.1.12.2 verResp? (ATT == ID, xml-lang, source, affiliation)
|    |         +---- 3.1.12.3 notes*   (ATT == ID, xml-lang, source, type, subject, level, resp,
|    |                                  sdatrefs)
|    |
|    |---- 3.2 locMap? (ATT == ID, xml-lang, source)
|    |    |
|    |    +---- 3.2.1 dataItem*    (ATT == ID, xml-lang, source, varRef, nCubeRef)
|    |         |
|    |         |---- 3.2.1.1 CubeCoord*(ATT == ID, xml-lang, source, coordNo, coordVal,
|    |         |                        coordValRef)
|    |         |                  [coordValRef is the varRef to the var where the coordVal is stored]
|    |         |
|    |         +---- 3.2.1.2 physLoc*    (ATT == ID, xml-lang, source, type, recRef, startPos,
|    |                                    width, endPos)
|    |
|    +---- 3.3 notes* (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|
|---- 4.0 dataDscr* (ATT == ID, xml-lang, source)
|    |
|    |---- 4.1 varGrp* (ATT == ID, xml-lang, source, type, var, varGrp, name, sdatrefs, methrefs,
|    |    |              pubrefs, access)
|    |    |---- 4.1.1 labl*     (ATT == ID, xml-lang, source, level, vendor, country, sdatrefs)
|    |    |---- 4.1.2 txt*      (ATT == ID, xml-lang, source, level, sdatrefs)
|    |    |---- 4.1.3 concept*  (ATT == ID, xml-lang, source, vocab, vocabURI)
|    |    |---- 4.1.4 defntn?   (ATT == ID, xml-lang, source)
|    |    |---- 4.1.5 universe? (ATT == ID, xml-lang, source, level, clusion)
|    |    +---- 4.1.6 notes*    (ATT == ID, xml-lang, source, type, subject, level, resp,
|    |                           sdatrefs)
|    |
|    |---- 4.2 nCubeGrp* (ATT == ID, xml-lang, source, type, nCube, nCubeGrp, name, sdatrefs, methrefs,
|    |    |               pubrefs, access)
|    |    |---- 4.2.1 labl*     (ATT == ID, xml-lang, source, level, vendor, country, sdatrefs)
|    |    |---- 4.2.2 txt*      (ATT == ID, xml-lang, source, level, sdatrefs)
|    |    |---- 4.2.3 concept*  (ATT == ID, xml-lang, source, vocab, vocabURI)
|    |    |---- 4.2.4 defntn?   (ATT == ID, xml-lang, source)
|    |    |---- 4.2.5 universe? (ATT == ID, xml-lang, source, level, clusion)
|    |    +---- 4.2.6 notes*    (ATT == ID, xml-lang, source, type, subject, level, resp,
|    |                           sdatrefs)
|    |
|    |---- 4.3 var* (ATT == ID, xml-lang, source, name, wgt, wgt-var, weight, qstn, files,
|    |    |          vendor, dcml, intrvl, rectype, sdatrefs, methrefs, pubrefs,
|    |    |          access, aggrMeth, measUnit, scale, origin, nature, additivity, temporal, geog,
|    |    |          geoVocab, catQnty)
|    |    |
|    |    |---- 4.3.1 location*    (ATT == ID, xml-lang, source, StartPos, EndPos, width,
|    |    |                         RecSegNo, fileid, locMap)
|    |    |---- 4.3.2 labl*       (ATT == ID, xml-lang, source, level, vendor, country, sdatrefs)
|    |    |---- 4.3.3 imputation?  (ATT == ID, xml-lang, source)
|    |    |---- 4.3.4 security?    (ATT == ID, xml-lang, source, date)
|    |    |---- 4.3.5 embargo?     (ATT == ID, xml-lang, source, date, event, format)
|    |    |---- 4.3.6 respUnit?    (ATT == ID, xml-lang, source)
|    |    |---- 4.3.7 anlysUnit?   (ATT == ID, xml-lang, source)
```

```
|   |   |---- 4.3.8 qstn*       (ATT == ID, xml-lang, source, qstn, var, seqNo, sdatrefs)
|   |   |   |
|   |   |   |---- 4.3.8.1 preQTxt*  (ATT == ID, xml-lang, source)
|   |   |   |---- 4.3.8.2 qstnLit*  (ATT == ID, xml-lang, source, sdatrefs)
|   |   |   |---- 4.3.8.3 postQTxt* (ATT == ID, xml-lang, source)
|   |   |   |---- 4.3.8.4 forward*  (ATT == ID, xml-lang, source, qstn)
|   |   |   |---- 4.3.8.5 backward* (ATT == ID, xml-lang, source, qstn)
|   |   |   +---- 4.3.8.6 ivuInstr* (ATT == ID, xml-lang, source)
|   |   |
|   |   |   NOTE: qstn ELEMENT may include mixed #PCDATA content.
|   |   |
|   |   |---- 4.3.9 valrng*       (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 4.3.9.1 range* (ATT == ID, xml-lang, source, UNITS, min, minExclusive
|   |   |   |                     max, maxExclusive)
|   |   |   |---- 4.3.9.2 item*  (ATT == ID, xml-lang, source, UNITS, VALUE)
|   |   |   |---- 4.3.9.3 key?   (ATT == ID, xml-lang, source)
|   |   |   +---- 4.3.9.4 notes* (ATT == ID, xml-lang, source, type, subject, level, resp,
|   |   |                         sdatrefs)
|   |   |
|   |   |---- 4.3.10 invalrng*     (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 4.3.10.1 range* (ATT == ID, xml-lang, source, UNITS, min, minExclusive
|   |   |   |                      max, maxExclusive)
|   |   |   |---- 4.3.10.2 item*  (ATT == ID, xml-lang, source, UNITS, VALUE)
|   |   |   |---- 4.3.10.3 key?   (ATT == ID, xml-lang, source)
|   |   |   +---- 4.3.10.4 notes* (ATT == ID, xml-lang, source, type, subject, level,
|   |   |                          resp, sdatrefs)
|   |   |
|   |   |---- 4.3.11 undocCod*   (ATT == ID, xml-lang, source)
|   |   |---- 4.3.12 universe*   (ATT == ID, xml-lang, source, level, clusion)
|   |   |---- 4.3.13 TotlResp?   (ATT == ID, xml-lang, source)
|   |   |---- 4.3.14 sumStat*    (ATT == ID, xml-lang, source, wgtd, wgt-var, weight, type)
|   |   |---- 4.3.15 txt*       (ATT == ID, xml-lang, source, level, sdatrefs)
|   |   |---- 4.3.16 stdCatgry*  (ATT == ID, xml-lang, source, URI)
|   |   |---- 4.3.17 catgryGrp*  (ATT == ID, xml-lang, source, missing, missType, catgry,
|   |   |   |                     catGrp, levelno, levelnm, compl, excls)
|   |   |   |
|   |   |   |---- 4.3.17.1 labl*    (ATT == ID, xml-lang, source, level, vendor, country, sdatrefs)
|   |   |   |---- 4.3.17.2 catStat* (ATT == ID, xml-lang, source, type, URI, sdatrefs, wgtd,
|   |   |   |                        wgt-var, weight)
|   |   |   +---- 4.3.17.3 txt*     (ATT == ID, xml-lang, source, level, sdatrefs)
|   |   |
|   |   |---- 4.3.18 catgry*     (ATT == ID, xml-lang, source, missing, missType, country,
|   |   |   |                     sdatrefs, excls)
|   |   |   |
|   |   |   |---- 4.3.18.1 catValu? (ATT == ID, xml-lang, source)
|   |   |   |---- 4.3.18.2 labl*    (ATT == ID, xml-lang, source, level, vendor, country, sdatrefs)
|   |   |   |---- 4.3.18.3 txt*     (ATT == ID, xml-lang, source, level, sdatrefs)
|   |   |   |---- 4.3.18.4 catStat* (ATT == ID, xml-lang, source, type, URI, sdatrefs, wgtd,
|   |   |   |                        wgt-var, weight)
|   |   |   +---- 4.3.18.5 mrow?      (ATT == ID, xml-lang, source)
|   |   |       |
|   |   |       +---- 4.3.18.5.1 mi*     (ATT == ID, xml-lang, source, varRef)
|   |   |
|   |   |---- 4.3.19 codInstr*   (ATT == ID, xml-lang, source)
|   |   |---- 4.3.20 verStmt*    (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 4.3.20.1 version? (ATT == ID, xml-lang, source, type, date)
|   |   |   |---- 4.3.20.2 verResp? (ATT == ID, xml-lang, source, affiliation)
|   |   |   +---- 4.3.20.3 notes*   (ATT == ID, xml-lang, source, type, subject, level,
```

```
|   |   |                          resp, sdatrefs)
|   |   |
|   |   |---- 4.3.21 concept*   (ATT == ID, xml-lang, source, vocab, vocabURI)
|   |   |---- 4.3.22 derivation? (ATT == ID, xml-lang, source, var)
|   |   |   |
|   |   |   |---- 4.3.22.1 drvdesc? (ATT == ID, xml-lang, source)
|   |   |   +---- 4.3.22.2 drvcmd?  (ATT == ID, xml-lang, source, syntax)
|   |   |
|   |   |---- 4.3.23 varFormat?  (ATT == ID, xml-lang, source, type, formatname, schema,
|   |   |                         category, URI)
|   |   |
|   |   |---- 4.3.24 geoMap* (ATT == ID, xml-lang, source, URI, mapformat, levelno)
|   |   +---- 4.3.25 notes*     (ATT == ID, xml-lang, source, type, subject, level, resp,
|   |                           sdatrefs)
|   |
|   |---- 4.4 nCube*     (ATT == ID, xml-lang, source, name, sdatrefs, methrefs, pubrefs,
|   |   |                 access, dmnsQnty, cellQnty)
|   |   |---- 4.4.1 location*   (ATT == ID, xml-lang, source, StartPos, EndPos, width,
|   |   |                        RecSegNo, fileid, locMap)
|   |   |---- 4.4.2 labl*       (ATT == ID, xml-lang, source, level, vendor, country, sdatrefs)
|   |   |---- 4.4.3 txt*        (ATT == ID, xml-lang, source, level, sdatrefs)
|   |   |---- 4.4.4 universe*   (ATT == ID, xml-lang, source, level, clusion)
|   |   |---- 4.4.5 imputation?  (ATT == ID, xml-lang, source)
|   |   |---- 4.4.6 security?   (ATT == ID, xml-lang, source, date)
|   |   |---- 4.4.7 embargo?    (ATT == ID, xml-lang, source, date, event, format)
|   |   |---- 4.4.8 respUnit?   (ATT == ID, xml-lang, source)
|   |   |---- 4.4.9 anlysUnit?  (ATT == ID, xml-lang, source)
|   |   |---- 4.4.10 verStmt*   (ATT == ID, xml-lang, source)
|   |   |   |
|   |   |   |---- 4.4.10.1 version? (ATT == ID, xml-lang, source, type, date)
|   |   |   |---- 4.4.10.2 verResp? (ATT == ID, xml-lang, source, affiliation)
|   |   |   +---- 4.4.10.3 notes*   (ATT == ID, xml-lang, source, type, subject, level,
|   |   |                            resp, sdatrefs)
|   |   |
|   |   |---- 4.4.11 purpose?   (ATT == ID, xml-lang, source, sdatrefs, methrefs, pubrefs, URI)
|   |   |
|   |   |---- 4.4.12 dmns*      (ATT == ID, xml-lang, source, rank, varRef)
|   |   |   |
|   |   |   +---- 4.4.12.1 cohort*   (ATT == ID, xml-lang, source, catRef, value)
|   |   |       |
|   |   |       +---- 4.4.12.1.1 range* (ATT == ID, xml-lang, source, UNITS, min, minExclusive
|   |   |                                max, maxExclusive)
|   |   |
|   |   |---- 4.4.13 measure*    (ATT == ID, xml-lang, source, varRef, aggrMeth, measUnit, scale,
|   |   |                         origin, additivity)
|   |   +---- 4.4.14 notes*     (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|   |
|   +---- 4.5 notes*     (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)
|
+---- 5.0 otherMat* (ATT == ID, xml-lang, source, type, level, URI)<--------------------------+
    |                                                                                         |
    |---- 5.1 labl* (ATT == ID, xml-lang, source, level, vendor, country, sdatrefs)         |
    |---- 5.2 txt?(ATT == ID, xml-lang, source, level, sdatrefs)                             |
    |---- 5.3 notes* (ATT == ID, xml-lang, source, type, subject, level, resp, sdatrefs)    |
    |---- 5.4 table* (ATT == ID, xml-lang, source)                                          |
    |---- 5.5 citation? (ATT == ID, xml-lang, source, MARCURI)                              |
    |         NOTE: full tree for citation element omitted for reasons of space.           |
    +---- 5.6 otherMat* (ATT == ID, xml-lang, source, type, level, URI) ------------------+
              NOTE: otherMat is recursively defined to 5.0.
```